บุญเลิศ เอี่ยมทัศนา

SOFT MAIL

เทอร์โบปาสคาล กับวินโดวส์
: หน้าต่างเอกสาร

เพื่อประสิทธิภาพที่สูงสุด ภายใต้การทำงานบนวินโดวส์ กับการใช้งานหน้าต่างเอกสาร

ในคู่มือวินโดวส์แบ่งหน้าต่างออกเป็น 2 ชนิด คือ หน้าต่างงาน (application windows) กับหน้าต่างเอกสาร (document windows) จากชนิดของหน้าต่างดังกล่าวมา หน้าต่างหลักคือหน้าต่างงาน แต่ยังไม่มีหน้าต่างชนิดที่ตรงกับคำนิยามของหน้าต่างเอกสาร ดังที่จะนำมากล่าวในตอนนี้

หน้าต่างเอกสาร

หน้าต่างเอกสารคือ หน้าต่างลูกของหน้าต่างงาน ซึ่งหน้าต่างงานคือหน้าต่างหลักดังกล่าวมา แต่ในเทอร์โบปาสคาลเรียกเป็นเฟรมวินโดว์ (frame window) คุณสมบัติที่สำคัญของหน้าต่างเอกสารคือ

- เป็นหน้าต่างที่มีแม่ (แม่คือหน้าต่างงาน)
- ซ้อนทับหน้าต่างแม่อยู่ตลอดเวลา และรวมทั้งไอคอนด้วยจะออกไปนอกพื้นที่ใช้งานของหน้าต่างแม่ไม่ได้
- มีส่วนประกอบของหน้าต่างเช่นเดียวกับหน้าต่างหลัก แต่ไม่มีเมนูบาร์เป็นของตนเอง ใช้ของหน้าต่างแม่ร่วมกัน
- จำนวนของหน้าต่างที่เปิดใช้งานขึ้นอยู่กับความต้องการของผู้ใช้
- จะต้องให้ผู้ใช้สามารถจัดเรียงหน้าต่างและไอคอนได้ (เช่น Tile)

กลุ่มสกุล MDI

คำว่า MDI ย่อมาจาก Multi Document Interface เพื่อการจัดการให้มีหน้าต่างทั้งสองแบบนี้ ซึ่งถ้าเขียนโปรแกรมด้วยภาษาซีจะต้องดำเนินการเอง แต่ในเทอร์โบปาสคาลได้ดำเนินการไว้แล้ว โดยสร้างเป็นสกุลชุดหนึ่งคือ

สกุลหน้าต่างงาน

คือสกุล TMDIWindow สืบสกุลจาก TWindow เพื่อที่จะดำเนินการกับหน้าต่างเอกสาร เทียบได้กับสกุล WndProC ในตัวอย่างที่ผ่านมา แต่สำหรับในกรณีนี้การเขียนโปรแกรมจะดำเนินการที่สกุลของหน้าต่างเอกสารมากกว่า เพราะการดำเนินการส่วนใหญ่จะอยู่ที่นั่น

สกุลหน้าต่างเงา

คือสกุล TMDIClient เป็นสกุลของหน้าต่างที่ไม่แสดงภาพหน้าต่างบนจอภาพ มีขอบเขตของหน้าต่างเต็มพื้นที่ใช้งานของหน้าต่างงาน มีลักษณะเหมือนหน้าต่างรองตามที่กล่าวในตอนก่อน

สืบสกุลจาก TControl (ที่จะกล่าวในตอนต่อไป) เป็นสกุลที่สร้างขึ้นเพื่อดำเนินการกับหน้าต่างเอกสารโดยตรง มีออปเจ็กต์ (พอยน์เตอร์ไปยังอินสแตนซ์) ของสกุลนี้อยู่ในสกุลหน้าต่างงาน เช่น เมื่อผู้ใช้เลือกรายการ Tile กิจกรรมในสกุลหน้าต่างงานจะเรียกใช้กิจกรรมในสกุลนี้ เพื่อจัดเรียงหน้าต่างให้อีกทอดหนึ่ง ปกติเราไม่ต้องยุ่งกับสกุลนี้ จึงของดกล่าวจนกว่าจะมีเรื่องต้องกล่าวถึงครับ

สกุลหน้าต่างเอกสาร

ใช้สกุล TWindow เป็นหน้าต่างเอกสาร แต่ในการเขียนโปรแกรมจะต้องสร้างสกุลหน้าต่างเอกสารขึ้นใหม่สืบสกุลจาก TWindow (หรือสกุลที่สืบสกุลจาก TWindow อีกทอดหนึ่งได้เช่นกัน) เพื่อให้ดำเนินการกับหน้าต่างเอกสารสกุลนี้เทียบได้กับสกุล SubWndC ตามที่กล่าวในตอนที่แล้ว

ตัวอย่างแรกของ MDI

การเขียนโปรแกรมเพื่อให้เกิดเป็นหน้าต่างงานกับหน้าต่างเอกสารจะดำเนินการได้ดังนี้คือ

- กำหนดสกุลของแอพพลิเคชันสืบสกุลจาก TApplication ตามปกติ

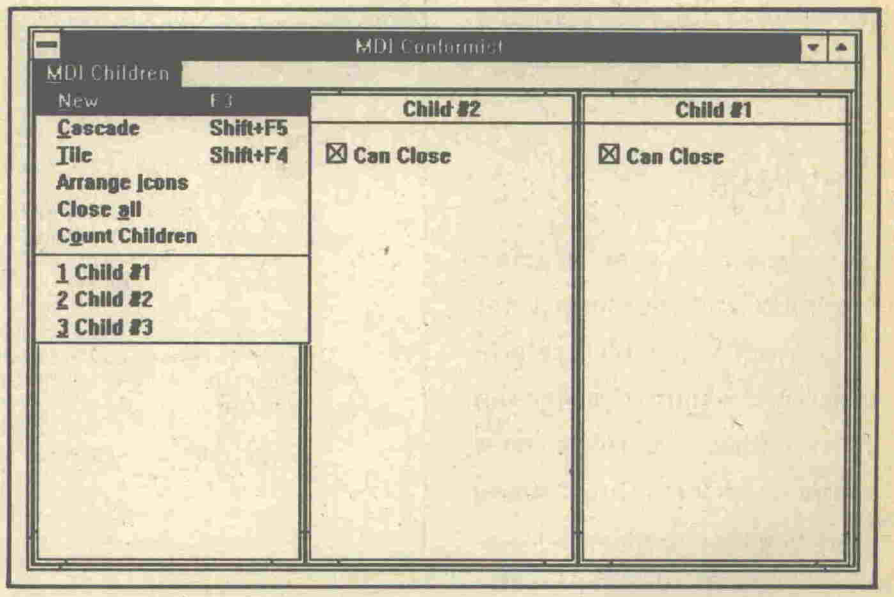
- กำหนดสกุล WndProC สืบสกุลจาก TMDIWindow เพื่อจัดการกับหน้าต่างเป็นส่วนรวม ซึ่งหากไม่มีการดำเนินการใดก็ไม่ต้องกำหนด

- กำหนดสกุล SubWndC สืบสกุลจาก TWindow (หรือจากผู้สืบสกุลอีกทอดหนึ่ง) เพื่อดำเนินการกับหน้าต่างเอกสาร ซึ่งหากไม่มีงานก็ไม่ต้องกำหนดเช่นกัน ก็จะได้เป็นโปรแกรมเปล่า คือใช้ทำอะไรไม่ได้ (นอกจากศึกษาลักษณะของหน้าต่างงานกับหน้าต่างเอกสาร) ดังโปรแกรม MDI ในรูปที่ 1

โปรแกรม MDI เป็นโปรแกรมตัวอย่างของเทอร์โบปาสคาล อยู่ในไฟล์ตามที่แสดงไว้ท้ายชื่อโปรแกรม เมื่อรันโปรแกรมนี้จะได้ผลการรันดังรูปที่ 2

```
program MDI;  { C:\TPW\OWLDEMOS\MDIAPP.PAS  }

{$R MDIAPP.RES}

uses WObjects, WinTypes, WinProcs;

type

  TMDIApp = object(TApplication)
    procedure InitMainWindow; virtual;
  end;


procedure TMDIApp.InitMainWindow;
begin
  MainWindow := New(PMDIWindow, Init('MDI Conformist',
    LoadMenu(HInstance, 'MDIMenu')));
end;


var
  MDIApp: TMDIApp;

begin
  MDIApp.Init('MDIApp');
  MDIApp.Run;
  MDIApp.Done;
end.
```

รูปที่ 1 โปรแกรม MDI



รูปที่ 2 ผลการรันโปรแกรม MDITest โดยเปิดหน้าต่าง 3 บาน แล้วให้จัดเรียงในแบบ Title กับให้แสดงเมนู

ซึ่งได้เปิดหน้าต่าง 3 บาน และให้จัดเรียงในแบบ Tile เป็นโปรแกรมเปล่าที่ไม่มีงานใด นอกจากจะทดลอง ย่อ ขยาย ย้าย มินิไมซ์และแมกซิไมซ์ หน้าต่างอันเป็นเรื่องของวินโดว์ กับการที่ให้เราเปิดใช้หน้าต่างได้ตามต้องการ สามารถจัดเรียงหน้าต่างและไอคอนได้ อันเป็นเรื่องที่เทอร์โบปาสคาลดำเนินการแทนเรา

ที่แตกต่างจากโปรแกรมที่เคยกล่าวมาคือ ใน InitMainWindow ได้ให้โหลดเมนูด้วย เพราะการใช้งานหน้าต่างเอกสารจะต้องมีเมนูดังแสดงในรูปที่ 2 (มีกิจกรรมสนองตอบรายการทั้งหมดนี้อยู่แล้วในสกุล TMDIWindow)

ทั้งหมดนี้คือที่ดำเนินการแทนเราไว้แล้ว ทำให้แบ่งเบาภาระในการเขียนโปรแกรมเป็นอย่างมาก เรื่องของเราคือการเขียนโปรแกรมดำเนินการกับหน้าต่างเหล่านี้

โดยผมจะนำไปใช้ในโครงงานห้องอาหาร ซึ่งเดิมมีโต๊ะอาหารอยู่ 50 ชุด แต่มีใบสั่งอาหารอยู่ใบเดียว เพราะใช้หน้าต่างหลักเป็นใบสั่งอาหาร แต่มีเรื่องอื่นต้องนำมาขัดจังหวะก่อนครับ เพราะผมยังไม่มีที่เก็บข้อมูลเพื่อเก็บใบสั่งอาหาร

ที่เก็บข้อมูล

ภาษาที่เป็นโอโอแอล หรือจะพูดให้ถูกคือในไลบรารีของคอมพายเลอร์ของภาษาเหล่านี้มักจะมีที่เก็บข้อมูลให้มาด้วย เพื่อให้เราเก็บข้อมูลได้ทุกชนิด ไม่ว่าจะเป็นสตริง อะเรย์หรือเรกอร์ด หรืออื่น ๆ (ทั้งนี้เพราะในที่เก็บจะเก็บแต่พอยน์เตอร์ของข้อมูลเท่านั้น)

การเก็บจะเก็บข้อมูลหลายชนิดรวมกันหรือแยกกันก็ได้ (ที่เก็บมีได้หลายที่) และเก็บได้ไม่จำกัดจำนวนจนกว่าจะเต็มหน่วยความจำ

ที่เก็บข้อมูลจะต้องมีกลไกคือโปรแกรมย่อยให้เรา นำข้อมูลไปเก็บ อ่านออกมาใช้งาน ถอนออกจากที่เก็บ เป็นต้น จึงเหมาะที่จะทำเป็นออปเจ็กต์ คือสร้างเป็นสกุลไว้และเช่นกันจึงนิยมใช้ในการเขียนโปรแกรมโอโอพี

เทอร์โบปาสคาลรุ่นก่อนไม่มีที่เก็บข้อมูลให้มา ในรุ่นนี้ได้ทำเป็นสกุล TCollection แต่ประสงค์ให้เก็บออปเจ็กต์ หากต้องการใช้เก็บข้อมูลโดด ๆ ต้องปรับปรุงกิจกรรมในสกุล TCollection หลายกิจกรรม

ทั้งนี้เพราะต้องการให้มีกิจกรรม Init กับ Done วิธีที่ง่ายจึงให้เก็บเป็นออปเจ็กต์โดยทำเป็นสกุล เช่นต้องการเก็บเลขจำนวนเต็มจะกำหนดสกุลได้ดังนี้

```
TYPE
     AgeC = OBJECT (TObject)
     age     Integer
             END;
```

ก็จะได้ Init และ Done ของ TObject (เป็นสกุลต้นตระกูลสุด) ให้ TCollection เรียกใช้

เตรียมออปเจ็กต์ให้ที่เก็บ

แต่เมื่อเขียนโปรแกรมจริง หากกำหนดเป็นออปเจ็กต์ก็อดไม่ได้ครับที่จะมีกิจกรรมอื่นอีก ดังเช่น ผมจะเก็บใบสั่งอาหารไว้ในที่เก็บ ใบสั่งอาหารของผม (โดยไม่นับหน้าต่าง) คืออะเรย์ของเลขจำนวนเต็มขนาด 21 ช่อง

แต่เมื่อทำเป็นออปเจ็กต์ จะกำหนดสกุลได้ดังรูป B1 และโปรแกรมตามรูป B2 ความหมายของผมคือ OdrPaperT เป็นใบสั่งอาหารที่เป็นกระดาษ คือ paper ขนาด 21 บรรทัด ซึ่งใบสั่งอาหารสามารถดำเนินกิจกรรมได้ดังนี้คือ

- Init บันทึกหมายเลขโต๊ะไว้ที่บรรทัดบน (ในอะเรย์ช่องแรก) ก่อนที่จะใช้ใบสั่งอาหาร

```
CONST  MaxDish = 20;

TYPE
  PaperT = array[0..MaxDish] of Integer;

  OdrPaperTP = ^OdrPaperT;
  OdrPaperT  = OBJECT(TObject)
    paper : PaperT;
    dcnt  : Integer;

  constructor  Init(tblnum : Integer);
  procedure    AddDish(dshnum : Integer);
  function     GetTblNum : Integer;
  function     GetDish(att : Integ r;
                 var dshnum : Integer) : Boolean;
  function     NumDish : Integer;
           END;
```

รูป B1

```
(* ----------------  OdrPaperT method ------------------ *)

CONSTRUCTOR  OdrPaperT.Init(tblnum : Integer);
BEGIN   TObject.Init;  paper[0] := tblnum;  dcnt := 0  END;


PROCEDURE  OdrPaperT.AddDish(dshnum : Integer);
BEGIN
  if  dcnt < MaxDish  then  inc(dcnt);
  paper[dcnt] := dshnum;
END;


FUNCTION  OdrPaperT.GetTblNum : Integer;
BEGIN  GetTblNum := paper[0]  END;


FUNCTION  OdrPaperT.GetDish(att : Integer;
                        var dshnum : Integer) : Boolean;
BEGIN
  if  (att > 0) and (att <= dcnt)  then
    begin  dshnum := paper[att];  GetDish := True  end
  else  GetDish := False;
END;


FUNCTION  OdrPaperT.NumDish : Integer;
BEGIN   NumDish := dcnt    END;
```

รูป B2

- AddDish บันทึกรายการอาหารที่ลูกค้าสั่ง โดยบันทึกเป็นหมายเลขในบรรทัดต่อ ๆ มา

- GetTblNum เพื่ออ่านหมายเลขโต๊ะที่บันทึกไว้

- GetDish เพื่ออ่านหมายเลขรายการอาหารที่บันทึกไว้

- NumDish เพื่อให้ทราบว่าบันทึกอาหารไปเป็นจำนวนจานเท่าไรแล้ว

เท่านี้ครับก็จะได้ใบสั่งอาหารที่พร้อมจะเก็บในที่เก็บข้อมูล ขอให้สังเกตว่าไม่มี Done เพราะไม่มีสิ่งใดที่จะต้องดำเนินการ หากใน Init จองหน่วยความจำไว้ก็จะต้องมี Done เพื่อปลดปล่อยหน่วยความจำ

แก้เมนู

เพื่อให้ใช้กิจกรรมที่ TMDIWindow เตรียมไว้ได้ คือ ตามที่เห็นในเมนูในรูปที่ 2 จะต้องจัดทำเมนูใหม่ให้เป็นดังที่แสดงในรูปที่ 3 โดยก็อบปี้ไฟล์ RESTRNT.RES ให้แก่ RSTRNT2.RES แล้วใช้ WRT ต่อเติมเมนูดังรูปที่ 3 ตอนล่าง

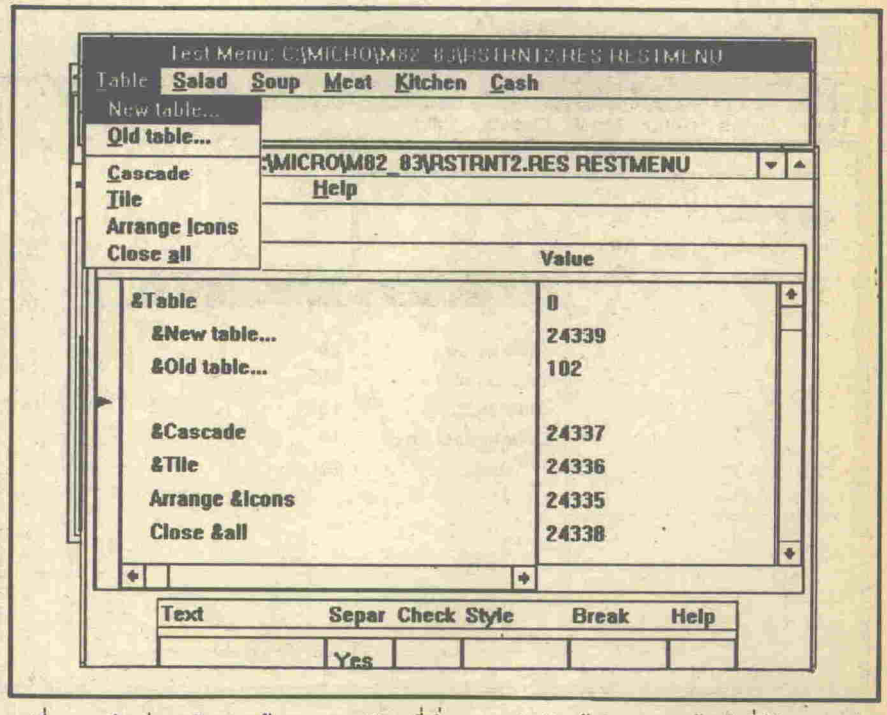
นำมาใช้ในโครงงาน

จากโปรแกรมของโครงงานร้านอาหารเดิม จะเห็นว่ามีใบสั่งอาหารเพียงใบเดียวสำหรับโต๊ะอาหาร 50 โต๊ะนำไปใช้งานไม่ได้ เพราะผมรอมาถึงตอนนี้ครับ

เพื่อจะนำหน้าต่างเอกสารแต่ละบานมาเป็นใบสั่งอาหาร คือมาเป็นกรอบของใบสั่งอาหารแต่ละใบ และจะใช้ที่เก็บข้อมูลเป็นที่เก็บใบสั่งอาหารด้วย (อาจออกแบบโดยไม่ต้องมีที่เก็บข้อมูลได้ ซึ่งจะขอนำไปก่อนในตอนท้าย ที่ให้มีเพื่อแสดงการเขียนโปรแกรมในหลายเรื่อง)

ดูผลการรันก่อน

เพื่อความเข้าใจง่ายผมคิดว่าดูผลการรันของโปรแกรมที่จะกล่าวถึงต่อไปนี้

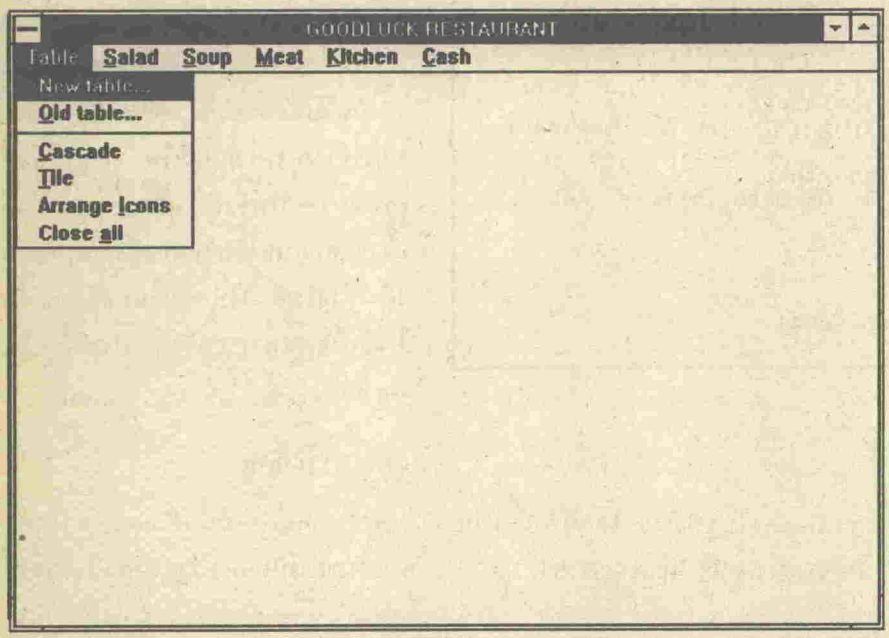


รูปที่ 3 เมนูรีซอร์สของโครงงานร้านอาหาร หลังจากที่เพิ่มรายการสำหรับหน้าต่างเอกสารแล้ว คือที่มีค่ารายการเป็นเลข 5 หลัก

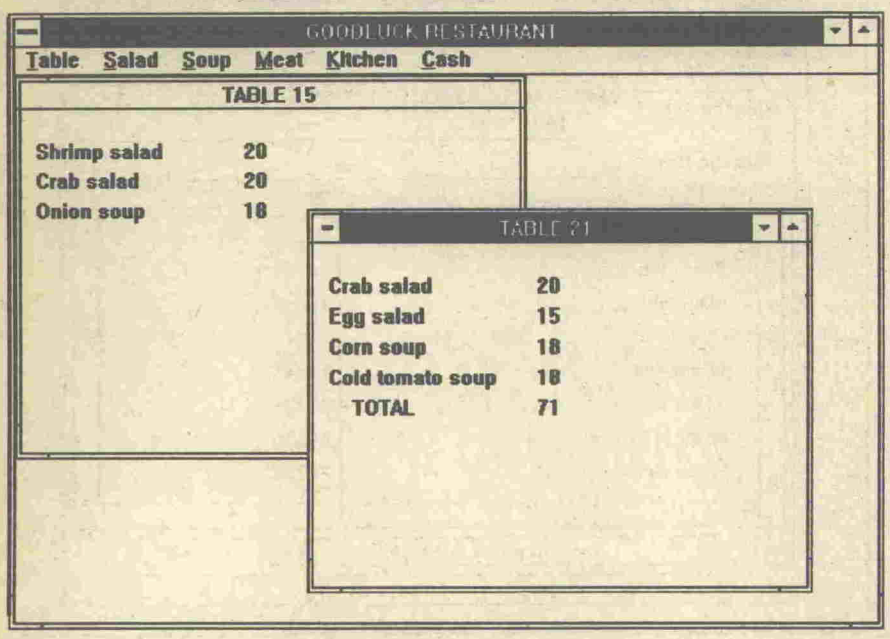
ก่อนดีกว่าครับ คือตามรูปที่ 4 เป็นผลการรันเมื่อเริ่มต้นมีแต่หน้าต่างงาน และผมให้แสดงเมนูของรายการ Table ให้ดูด้วย

เมื่อผู้ใช้เลือกรายการ Table/New table และป้อนหมายเลขโต๊ะแล้ว ก็จะเป็นการเปิดหน้าต่างเอกสาร โดยผมนำหมายเลขโต๊ะมาเป็นชื่อหน้าต่างดังแสดงในรูปที่ 5 แต่ตามรูปนี้ผมได้เปิดหน้าต่างเอกสาร 2 บาน เพื่อแสดงว่าเปิดได้หลายบาน

งานต่อไปของผู้ใช้ที่เป็นเจ้าหน้าที่ร้านอาหาร คือ บันทึกรายการอาหารที่ลูกค้าสั่งโดยเลือกจากเมนู ดังแสดงในรูปที่ 5 เช่นกัน และเมื่อลูกค้าชำระเงิน เจ้าหน้าที่ละเลือกรายการ Cash ซึ่งจะแสดงยอดรวม (TOTAL) ดังแสดงในรูปที่ 5 อีกเช่นกันครับ



รูปที่ 4 ผลการรันของโปรแกรม RestaurantTwo เมื่อเริ่มต้น



รูปที่ 5 ผลการรันของโปรแกรม RestaurantTwo เมื่อได้เปิดใช้หน้าต่างเอกสารแล้ว มีการบันทึกรายการอาหารและคิดยอดเงินรวม

เป็นการจบการทำงานของใบสั่งอาหารหนึ่ง ๆ ซึ่งในโปรแกรมผมให้ฉีกทิ้ง คือถอนจากที่เก็บข้อมูลและยกเลิกหน่วยความจำ แต่ตัวหน้าต่างคงอยู่เป็นหน้าต่างเปล่า คือหน้าต่างที่ไม่มีใบสั่งอาหาร ในคำอธิบายการใช้โปรแกรมจะให้ผู้ใช้ปิด แต่ถ้าไม่ปิดก็คงจะเกะกะจอภาพต่อไป จนกว่าจะมีลูกค้ามาใช้โต๊ะนั้นอีก

ทางเดินของสื่อความ

อีกเรื่องหนึ่งที่ผมขอทำความเข้าใจไว้ก่อนคือ ขอให้ดูที่เมนูในรูปที่ 4 บางรายการเช่น รายการใน Table ทั้งหมดจะไปทำงานที่ (โปรแกรมของ) หน้าต่างแม่ บางรายการเช่น รายการใน Salad, Soup และ Cash จะไปทำงานที่ (โปรแกรมของ) หน้าต่างลูก

ที่เป็นเช่นนี้เพราะเมื่อผู้ใช้เลือกรายการใดก็ตาม วินโดวส์จะส่งสื่อความ (ตามคำที่เรากำหนดไว้ในรีซอร์ส) ให้แก่หน้าต่างลูก หากหน้าต่างลูกไม่รับคือไม่ได้จัดกิจกรรมสนองตอบสื่อความนั้น ก็จะส่งต่อให้แก่หน้าต่างแม่

ด้วยวิธีนี้จึงทำให้สามารถกำหนดได้ว่า จะให้ใครแม่หรือลูกดำเนินการใด (สำหรับ Meat และ Kichen ที่ไม่กล่าวถึง เป็นเรื่องของตอนต่อไปครับ)

เตรียมโปรแกรม

ขอกลับไปที่เรื่องของโปรแกรม ซึ่งจะต้องจัดเตรียมโปรแกรมเก่าเป็นโปรแกรมใหม่ก่อนครับ ขอให้ก๊อบปี้ไฟล์

RESTRNT.PAS เดิม (ในไมโครฯ ฉบับที่ 82) เป็นไฟล์ RSTRNT2.PAS และเมื่ออยู่เทอร์โบปาสคาลแล้วขอให้แก้โปรแกรมบางส่วนตามคำอธิบายและตามรูป B3

สร้างสกุลหน้าต่างเอกสาร

ตามเนื้อเรื่องแล้วผมควรต้องกล่าวถึงสกุลหน้าต่างงานก่อน แต่เนื่องจากต้องแก้ชื่อสกุลจึงต้องนำหน้าต่างเอกสารมากล่าวก่อนคือ ตามโปรแกรมเดิมใช้พื้นที่ในหน้าต่างหลักเป็นใบสั่งอาหาร การดำเนินการต่าง ๆ จึงอยู่ในโปรแกรมของหน้าต่างหลักคือในสกุล WndProC แต่ในโปรแกรมใหม่นี้ต้องการใช้พื้นที่ของหน้าต่างเอกสารเป็นใบสั่งอาหาร จึงต้องย้ายการดำเนินการดังกล่าวมาไว้ในโปรแกรมของสกุลหน้าต่างเอกสาร ซึ่งจะให้ชื่อว่า SubWndC

```
PROGRAM  RestaurantTwo;

{$R RSTRNT2.RES}

USES  Strings,WinTypes,WinProcs,WObjects,StdDlgs;

TYPE   dishnmeT = array[0..30] of Char;

CONST
  cm_NewTbl     = 101;      cm_OldTbl     = 102;
     ..........................
  cm_Cash       = 50;       cm_Kitchen    = 51;
  wm_Cashing    = wm_User;

  dish : array[1..14] of dishnmeT =
    ('Shrimp salad       ', 'Crab salad         ',
     ..........................
     'Sirloin steak      ', 'Peper steak        ');

   price : array[1..14] of Integer =
     (20,20,20,15,15,18,18,18,18,70,60,80,60,50);
```

รูป B3 ส่วนของโปรแกรมสำหรับการแก้โปรแกรมเดิมคือ
- แก้ชื่อโปรแกรม
- แก้ชื่อไฟล์ .RES
- เพิ่มค่าคงตัว wm_Cashing
- เปลี่ยนค่าคงตัวใน price จาก 12 เป็น 18

```
SubWndC = OBJECT(TWindow)
  slip : OdrPaperTP;
  tblnum,row : Integer;

constructor  Init(apar : PWindowsObject; aslp : OdrPaperTP);
procedure  Paint(PaintDC : HDC;
                 var PaintInfo : TPaintStruct); virtual;
procedure  NewTable(nslp : OdrPaperTP);
procedure  cmOldTbl(var msg : TMessage);
             virtual cm_First + cm_OldSlip;
.....................................................
             END;
```

รูป B4 ส่วนของโปรแกรมสำหรับแก้โปรแกรมเดิมหลังจากเปลี่ยนชื่อสกุลแล้ว คือ
- ตัดกิจกรรม cmNewTbl และตัวโปรแกรม
- เพิ่มกิจกรรม NewTable

วิธีการง่ายมากครับคือ ให้เปลี่ยนชื่อสกุลจากของเดิมคือ WndProC เป็น SubWndC และเปลี่ยน WndProCP เป็น SubWndCP ทุก ๆ แห่ง (ใช้ Search/Replace) จะได้การกำหนดสกุลใหม่นี้ตามรูป B4 ขอให้แก้โปรแกรมตามที่กล่าวในรูปนี้ด้วย เป็นการเตรียมโปรแกรมเพื่อต่อเติมต่อไปครับ

ในสกุลใหม่นี้มีตัวแปร slip คือใบสั่งอาหารตามที่กำหนดไว้เป็นสกุล OrdPaperT แต่ให้เป็นพอยน์เตอร์ไปยังใบสั่งอาหารซึ่งเก็บอยู่ในที่เก็บข้อมูล เป็นวิธีใช้ข้อมูลในที่เก็บข้อมูลวิธีหนึ่งครับ

ส่วนตัวแปร tblnum คือหมายเลขโต๊ะที่อยู่ในใบสั่งอาหาร อ่านมาไว้ในตัวแปรนี้เพื่อความสะดวกในการเขียนโปรแกรม และคงต้องขอพักเรื่องของหน้าต่างเอกสารไว้เท่านี้ก่อนครับ

สกุลของหน้าต่างงาน

ในโครงงานนี้จำเป็นต้องสร้างสกุลหน้าต่างงาน คือ WndProC สืบสกุลจาก TMDIWindow เพราะจะต้องกำหนดที่เก็บข้อมูลเมื่อเริ่มต้น กำเนิดใบสั่งอาหารแต่ละใบเพื่อมอบ (แอดเดรส) ให้แก่หน้าต่างเอกสาร จึงต้องแย่งการกำเนิดหน้าต่างเอกสารจากสกุล TMDIWindow มาดำเนินการ

คำว่ากำเนิดคือ create ครับ มองหาคำที่เหมาะสมกว่านี้ไม่ได้ จะใช้คำว่าสร้างก็ไม่ใช่ เช่น การสร้างหน้าต่างดำเนินการมาก่อนจุดนี้เป็นจุดให้กำเนิดคือทำให้เกิดขึ้นครับ จะได้สกุลตามรูป B5 มีสกุลของแอพพลิเคชัน คือ ResturntC รวมอยู่ด้วย และได้กำหนด SubWndCP ไว้ก่อนสกุล WndProC ด้วย เพราะมีการอ้างถึงใน WndProC คือที่กิจกรรม OldWnd แต่ผมป้อน

โปรแกรมตามรูป B4 ไว้หลังโปรแกรมตามรูป B5

นำที่เก็บข้อมูลมาใช้

สำหรับในสกุล WndProC ตัวแปร pole คือที่เก็บข้อมูลครับ ผมนึกถึงตามร้านอาหารที่มักจะนำใบสั่งอาหารเสียบไว้ที่ที่เสียบ

เมื่อนำมาใช้ก็ต้องให้กำเนิดดังใน Init ในรูป B6 คือที่ให้ New PCollection โดยที่ที่เก็บข้อมูลเป็นอะเรย์ค่าที่ให้เมื่อ Init คือ 30 เป็นจำนวนช่องต่ำสุด ส่วนค่า 10 คือค่าที่ให้ขยายได้ในแต่ละครั้งเมื่อที่เก็บเต็ม

เหมือนกันอย่างประหลาดครับคือระหว่างที่ผมคอยให้ในเทอร์โบปาสคาลมีที่เก็บข้อมูล ผมต้องสร้างของผมขึ้นมาก่อนไม่เช่นนั้นโปรแกรมจะไม่เป็นโอโอพี โดยผมเลี่ยงไปใช้ชื่อว่า Box เพราะคาดว่าในเทอร์โบปาสคาลคงใช้ชื่อว่า Collection ตามชื่อในภาษาสมอลล์-ทอล์ค (Smalltalk)

ที่ว่าเหมือนกันคือ ใช้อะเรย์ไม่จำกัดขนาดคืออะเรย์ในฮีพเหมือนกัน แทนที่จะใช้ลิงค์ลิสต์ ให้บ่งขนาดจำนวนช่องต่ำสุดเหมือนกัน แต่ของผมให้ขยายโดยอัตโนมัติครั้งละ 25 เปอร์เซ็นต์ ดูก็ไม่มีสิ่งใดมาก แต่ผมต้องใช้เวลาตกลงใจในแต่ละเรื่องนานพอสมควร

ตัวเลข 30 ผมนำมาจาก สมมติว่าสอบถามผู้ใช้และได้ได้คำตอบว่าลูกค้าโดยปกติประมาณ 20 ถึง 30 โต๊ะ แต่ถ้ากำหนดขนาดของที่เก็บไว้ใหญ่เกินไปก็จะเปลืองหน่วยความจำโดยใช่เหตุ แต่ถ้าเล็กเกินไปก็จะต้องเสียเวลาในการขยายที่เก็บแต่ละครั้ง แต่ความจริงแล้วในกรณีนี้ควรให้เป็น 50 จะเหมาะสมกว่า เนื่องจากที่เก็บช่องหนึ่งใช้เนื้อที่เพียง 4 ไบต์เท่านั้น จึงควรกำหนดให้ใหญ่สุดไว้ เพราะเวลาที่ใช้ในการขยายที่เก็บสำคัญกว่า

ในรูป B5 มีโปรแกรมของกิจกรรม Done อยู่ด้วย เพื่อยกเลิกหน่วยความจำที่จองไว้เป็นที่เก็บข้อมูลเมื่อจะเลิกใช้ ในกรณีนี้คือเมื่อผู้ใช้จะเลิกใช้ ในกรณีนี้คือเมื่อผู้ใช้ยุติการรันโปรแกรมนี้ ซึ่งถ้าเป็นบนดอสหากไม่ยกเลิกไม่มีปัญหาใด

```
TYPE
  ResTurntC = OBJECT(TApplication)
    procedure  InitMainWindow; virtual;
            END;

  SubWndCP =  SubWndC;

  WndProCP  = ^WndProC;
  WndProC   = OBJECT(TMDIWindow)
    pole  : PCollection;

  constructor  Init(atitle : PChar; amenu : HMenu);
  destructor   Done; virtual;
  function     CreateChild : PWindowsObject; virtual;
  procedure    cmCashing(var msg : TMessage);
                 virtual wm_First + wm_Cashing;
  function     OldWnd(tblnum : Integer;
                      var wndptr : SubWndCP) : Boolean;
  function     OldSlip(tblnum : Integer;
                       var slpptr : OdrPaperTP) : Boolean;
  function     EntTblNum(var tblnum : Integer;
                         tblstr : PChar) : Boolean;
            END;
```

รูป B5

```
(* ---------------------- WndProc method ---------------- *)

CONSTRUCTOR  WndProC.Init(atitle : PChar; amenu : HMenu);
BEGIN
  TMDIWindow.Init(atitle,amenu);
  pole := New(PCollection, Init(30,10));
END;

DESTRUCTOR  WndProC.Done;
BEGIN   Dispose(pole);  TMDIWindow.Done  END;
```

รูป B6

สถานะของหน้าต่างเอกสาร

เมื่อผู้ใช้เปิดหน้าต่างเอกสารแล้วอาจดำเนินการกับหน้าต่างได้หลายอย่าง เช่น ย่อ ขยาย ย้าย และจัดเรียง เป็นต้น ที่จะส่งผลกระทบมาถึงโปรแกรมเราคือ

- เปิดหน้าต่างมาซ้อนทับกัน
- มินิไมซ์เป็นไอคอน
- ปิดหน้าต่าง คือ Close (แต่ใบสั่งอาหารยังคงอยู่เพราะเก็บในที่เก็บ)
- เป็นหน้าต่างเปล่า (ไม่มีใบสั่งอาหาร เกิดจากเลือกรายการ Cash แล้วไม่ได้ปิดหน้าต่าง)

ให้บริการผู้ใช้

เมื่อหน้าต่างซ้อนทับกันหากผู้ใช้ต้องการจะดำเนินการกับหน้าต่างใดก็ต้องนำหน้าต่างนั้นมาไว้ชั้นบนสุด ซึ่งจะ

ทำได้โดยคลิ๊กที่หน้าต่างนั้น (หากไม่ถูกทับทั้งหมด) กดคีย์ Crtl + tab วนหาหน้าต่างที่ต้องการ หรือเลือกในรายการ Table (ดังเช่นในรูปที่ 2) ซึ่งถ้ามีหน้าต่างเป็นจำนวนมากจะแสดงไดอะล็อกบ็อกซ์ให้เลือกดังแสดงในรูปที่ 6

ทั้งหมดนี้หากมีหน้าต่างเป็นจำนวนมาก เช่น เมื่อมีลูกค้ามากกว่า 20 โต๊ะ จะทำได้ยากพอสมควร ควรจะมีหนทางอื่นให้ผู้ใช้เลือกอีก ก็คงต้องเป็นแบบเก่าคือให้ป้อนหมายเลขโต๊ะ ปัญหาคือจะให้เลือกจากรายการใด

ที่ถูกควรใช้รายการ Old table โดยให้อยู่ที่เมนูบาร์ แต่ผมต้องการใช้รายการ Table/New table (ที่ถูกควรให้ชื่อเป็น Table/Table โดยไม่มี Old table) เพื่อดำเนินการทั้งปวงในการเปิดใช้หน้าต่างเอกสาร ดังนั้นรายการนี้จึงไม่เพียงแต่เปิดหน้าต่างใหม่เท่านั้น หากให้เปิดใช้หน้าต่างเก่าหรือทำให้แอคตีฟด้วย

เปิดใช้หน้าต่างเอกสาร

เมื่อเป็นเช่นนี้ผู้ใช้จะเลือกรายการ Table/New table โดยไม่คำนึงว่าหน้าต่างนั้นจะเปิดอยู่ (แอคตีฟไม่แอคตีฟ หรือเป็นไอคอน) หรือไม่มีหน้าต่างนั้น

ในด้านการทำงาน เมื่อผู้ใช้เลือกรายการ Table/New table วินโดว์จะส่งสื่อความ cm_CreateChile (ค่า 24339 ตามที่ให้ไว้ในรีซอร์ส) มีกิจกรรมสนองตอบสื่อความนี้อยู่ใน TMDIWindow แล้ว แต่ทำให้ปรับปรุงได้โดยเรียกใช้ CreateChild ซึ่งเป็นเวอร์ชวล

ผมได้กำหนดกิจกรรมทับในรูป B5 และนำมาปรับปรุงตามรูป B7 ขอให้ดูตรงตัวโปรแกรมหลักของ CreateChile คือตอนล่างของรูปก่อนครับ เริ่มจากให้

รูปที่ 6 ไดอะล็อกบ็อกซ์แสดงรายชื่อหน้าต่างให้เลือก เมื่อเปิดหน้าต่างเอกสารมากกว่า 9 บาน

ผู้ใช้ป้อนหมายเลขโต๊ะด้วย EntTblNum ได้มาใน tnum

เมื่อผู้ใช้ป้อนแล้วผมได้นำหมายเลขโต๊ะมาตรวจว่าเป็นหน้าต่างเก่าคือ เปิดอยู่หรือไม่ ด้วย OldWnd และเป็นใบสั่งเก่าคือ ที่อยู่ในที่เก็บหรือไม่ด้วย OldSilp นำเงื่อนไขมาดำเนินการดังนี้ (ข้อความในวงเล็บคือสาเหตุ)

1. ใบสั่งอาหารเก่า (ลูกค้าเก่า)

1.1 หน้าต่างเก่า (อาจถูกบังอยู่) ให้เปิดหน้าต่างเก่าด้วย PopSubWnd

1.2 หน้าต่างใหม่ (ผู้ใช้ปิดหน้าต่าง) ให้เปิดหน้าต่างใหม่ด้วย CreateNew

2. ใบสั่งอาหารใหม่ (ลูกค้ารายใหม่) จะให้กำเนิดใบสั่งอาหารใหม่ด้วย New แล้วนำไปเก็บในที่เก็บข้อมูลด้วย Insert ขอให้สังเกตตัวแปร sptr ไว้ด้วยครับ เป็นพอยน์เตอร์ไปยังหน่วยความจำในยีพที่เป็นใบสั่งอาหาร แล้วดำเนินการต่อคือ ถ้าเป็น

```
FUNCTION  WndProC.CreateChild : PWindowsObject;
VAR  tnum : Integer;  tstr : array[0..5] of Char;
     wptr : SubWndCP;  sptr : OdrPaperTP;  ownd : Boolean;

  procedure  CreateNew;
  var  tem : Pointer;
  begin  tem := Application^.MakeWindow(New(SubWndCP,
                          Init(@self,sptr)));
         wptr := tem;  end;

  procedure  PopSubWnd;
  begin
    with  wptr^  do
      if  IsIconic(HWindow)  then  OpenIcon(Hwindow)
      else  SetFocus(HWindow);
  end;


  procedure  WndNewSlip;
  begin  wptr^.NewTable(sptr);  PopSubWnd;
         InvalidateRect(wptr^.Hwindow,nil,true)  end;

BEGIN     { CreateChild  }
  if  EntTblNum(tnum, tstr)  then
    begin
    ownd := OldWnd(tnum,wptr);
    if  OldSlip(tnum,sptr)  then
      if  ownd  then  PopSubWnd  else  CreateNew
    else
      begin
      New(sptr,Init(tnum));  pole^.Insert(sptr);
      if  ownd  then  WndNewSlip  else  CreateNew;
      end;
    CreateChild := wptr;
    end
  else  CreateChild := nil;
END;
```

รูป B7

2.1 หน้าต่างเก่า (เลือกรายการ Cash แล้วไม่ปิดหน้าต่างของโต๊ะนี้) ให้ใบสั่งอาหารใหม่ด้วย WndNewSlip

2.2 หน้าต่างใหม่ (ยังไม่เคยเปิดหน้าต่างสำหรับโต๊ะนี้เลย หรือเปิดแล้วปิดแล้ว) ให้เปิดหน้าต่างใหม่ด้วย CreateNew

สำหรับการทำงานของโปรแกรมย่อยที่เป็นโลคัลของ CreateChild มีดังนี้ครับ

* CreateNew ให้กำเนิดหน้าต่างใหม่ โดยที่

- เมื่อ New จะจองเนื้อที่สำหรับหน้าต่างให้ค่าพอยน์เตอร์แก่ตัวแปรที่เป็นอาร์กิวเมนต์ของ MakeWindow ซึ่งจะให้ค่านี้แก่ MakeWindow อีกทอดหนึ่ง ผมนำมาให้แก่ Tem เพื่อถ่ายทอดให้แก่ wptr (หมายถึง พอยน์เตอร์ของหน้าต่าง)

- เมื่อ New ได้ให้ Init หน้าต่างเอกสารด้วย โดยที่ self หมายถึงให้หน้าต่างงานเป็นหน้าต่างแม่

- สำหรับค่า sptr คือ ใบสั่งอาหาร ซึ่งถ้าเป็นตามข้อ 1.2 จะเป็นใบสั่งเก่าในที่เก็บตามที่ OldSlip อ่านมาให้ แต่ถ้าเป็นตามข้อ 2.2 จะเป็นใบสั่งใหม่ที่เพิ่งกำหนด

* PopSubWnd เปิดใช้หน้าต่างเก่าโดยให้ตรวจว่าเป็นไอคอนหรือไม่ด้วย IsIconic ซึ่ง

- ถ้าเป็นไอคอน ให้เปิดหน้าต่างจากไอคอนด้วย OpenIcon

- หากไม่ใช่ให้ถือว่าไม่แอคตีฟ (ถึงแม้จะแอคตีฟอยู่ก็ไม่มีปัญหา) ให้นำขึ้นชั้นบนและแอคตีฟด้วย SetFocus

สำหรับค่า HWindow ในโปรแกรมย่อยนี้เป็นของหน้าต่างเอกสาร คือ wptrâ.HWindow

```
FUNCTION  WndProC.OldSlip(tblnum : Integer;
                  var slpptr : OdrPaperTP) : Boolean;

  function  TestTbl(tbl : OdrPaperTP) : Boolean; far;
  begin  TestTbl := tblnum = tbl^.GetTblNum   end;

BEGIN
  slpptr := pole^.FirstThat(@TestTbl);
  OldSlip := slpptr <> nil;
END;


FUNCTION  WndProC.OldWnd(tblnum : Integer;
                        var wndptr : SubWndCP) : Boolean;

  function  TestWnd(wnd : SubWndCP) : Boolean; far;
  begin
    if tblnum = wnd^.tblnum  then
      begin  wndptr := wnd;  TestWnd := True  end
    else  TestWnd := False;
  end;

BEGIN
  OldWnd := FirstThat(@TestWnd) <> nil;
END;


FUNCTION  WndProC.EntTblNum(var tblnum : Integer;
                          tblstr : PChar) : Boolean;
VAR   dc : HDC;  res : Integer;
BEGIN
  tblstr := '0';
  if  Application^.ExecDialog( New(PInputDialog,
      Init(@self,'Table number','From 1 to 50',
           tblstr,SizeOf(tblstr)))) = id_OK  then
    begin
    Val(tblstr,tblnum,res);
    EntTblNum := res = 0;
    end
  else  EntTblNum := False;
END;
```

รูป B8

* WndNewSlip ให้ใบสั่งอาหาร (ใหม่) แก่หน้าต่าง (เก่า) โดย

- ให้ใบสั่งอาหารใหม่ด้วย SubWndC.NewTable

- หน้าต่างอาจไม่แอคตีฟ จึงให้แอคตีฟด้วย PopSubWnd

- ลบภาพในหน้าต่างด้วย InvalidateRect โดยที่ nil หมายถึงทั้งหน้าต่าง และ true คือให้ลบ

FirstThat

กลไกสำคัญอันหนึ่งของที่เก็บข้อมูลคือ กิจกรรม FirstThat ซึ่งจะค้นหาข้อมูลตามที่ต้องการในที่เก็บให้ ดังที่ผมนำมาใช้ในกิจกรรม OldSlip ในรูป B8 การใช้งาน FirstThat มีข้อกำหนดซึ่งผมจะนำ OldSlip มาประกอบคำอธิบายด้วยคือ

1. ให้จัดกิจกรรม (หรือโปรแกรมย่อย) เป็นผู้ใช้ FirstThat จะเป็นโพรซีเดอร์หรือฟังก์ชันก็ได้ ตามตัวอย่างนี้คือ Oldslip

2. ให้จัดฟังก์ชันตรวจหาข้อมูลตามตัวอย่างคือ TestTb1 ซึ่งจะต้อง

2.1 เป็นฟังก์ชันแบบบูลีน ให้

ค่าความเป็นจริงเมื่อตรวจข้อมูลพบ และเป็นเท็จเมื่อไม่พบ

2.2 มีอาร์กิวเมนต์เป็นพอยน์เตอร์ของแบบข้อมูลที่เก็บอยู่ในที่เก็บข้อมูลเพียงอาร์กิวเมนต์เดียว

2.3 เป็นโลคัลคือเป็นฟังก์ชันภายในของกิจกรรม

2.4 คอมพายล์ในแบบ FAR คือด้วยไดเรคตีฟ {$F+} หรือ far

3. ให้ผ่านค่าฟังก์ชันนี้เป็นพารามิเตอร์เมื่อเรียกใช้ FirstThat

เมื่อดำเนินการตามที่กล่าวมา FirstThat จะดำเนินการดังนี้ คือ

- ให้ค่าออปเจ็กต์ (พอยน์เตอร์ไปยังออปเจ็กต์) เริ่มจากต้นลิสต์คือต้นอะเรย์แก่ฟังก์ชันตรวจหาข้อมูล คือ TestTbl

- ถ้าฟังก์ชันให้ค่าเป็นเท็จ FirstThat จะให้ค่าออปเจ็กต์ต่อไป และต่อ ๆ ไปอีกหากยังเป็นเท็จจนหมดข้อมูลในที่เก็บ FirstThat จะให้ตนเองเป็น nil กลับไป

- หากฟังก์ชันให้ค่าเป็นจริง FirstThat จะให้ค่าออปเจ็กต์แก่ตนเองกลับไป

ผมจึงนำมาค้นหาใบสั่งอาหารได้ดังที่เห็นในโปรแกรม

ใช้งานลิงค์ลิสต์

ในหน้าต่างทุกบานมีตัวแปร ChildList อยู่ เป็นพอยน์เตอร์เพื่อจะเก็บหน้าต่างลูกเป็นลิงค์ลิสต์ คือเมื่อเราใช้กิจกรรม MakeWindow ดังใน CreateNew (รูป B7) จะเก็บหน้าต่างใหม่นี้ลงลิงค์ลิสต์ด้วย แต่ไม่ค่อยได้ใช้งานลิงค์ลิสต์กัน

สามารถใช้งานลิงค์ลิสต์ได้ด้วย FirstThat เช่นกันทำหน้าที่เดียวกัน มีข้อกำหนดเหมือนกัน ชื่อเหมือนกัน แต่ไม่ใช่กิจกรรมเดียวกันของลิงค์ลิสต์อยู่ในสกุล TWindowsObject

ดังที่ผมทำมาใช้ในกิจกรรม OldWnd เพื่อค้นหาหน้าต่างเก่า โดยตรวจจากหมายเลขโต๊ะ โปรแกรมอยู่ในรูป B8 เช่นกัน

อินิตของหน้าต่างเอกสาร

เมื่อ WndProC.CreateChild (รูป B7) ให้กำเนิดหน้าต่างเอกสารคือใน CreateNew อีกทอดหนึ่ง ได้เรียกใช้ Init ของ SubWndC และผ่านค่าออปเจ็กต์ของตนกับของใบสั่งอาหารมาให้แก่ตัวแปร apar และ aslp ที่ส่วนหัวของ Init ตามลำดับ คือในรูป B9

โดยที่ apar หมายถึงหน้าต่างแม่ และ aslp หมายถึงใบสั่งอาหาร โดย Init จะอ่านหมายเลขโต๊ะจาก aslp นำมาทำเป็นชื่อหน้าต่าง และส่ง apar กับหน้าต่างไปให้แก่ TWindow.Init ซึ่งจะนำ apar ไปเก็บไว้ในตัวแปร Parent (โดย Init ของ TWindowsObject)

การดำเนินการเช่นนี้จะทำให้หน้าต่างลูกทุกบานรู้จักแม่ของตน ดังจะเห็นการใช้ต่อไป สำหรับค่าใน aslp ได้นำไปเก็บไว้ในตัวแปร slip ของสกุลนี้ เพื่อใช้ในการบันทึกรายการอาหาร

ในรูปเดียวกันนี้ (B9) มีกิจกรรม Paint ที่แก้ใหม่ ซึ่งขอให้สังเกตด้วยว่าถ้า slip เป็น nil คือไม่มีใบสั่งอาหาร จะไม่วาดภาพบนจอภาพใหม่

อีกกิจกรรมหนึ่งคือ NewTable เป็นกิจกรรมใหม่ เพื่อให้ใบสั่งอาหารใบใหม่แก่ slip

```
(* ------------------  SubWndC method  ---------------- *)

CONSTRUCTOR SubWndC.Init(apar : PWindowsObject;
                         aslp : OdrPaperTP);
VAR   atitle : array[0..12] of Char;
      wnme   : array[0..5] of Char;
BEGIN
  tblnum := aslp^.GetTblNum;  Str(tblnum,wnme);
  StrCat(StrECopy(atitle,'TABLE '),wnme);
  TWindow.Init(apar, atitle);
  slip := aslp;
END;


PROCEDURE  SubWndC.Paint(PaintDC : HDC;
                         var PaintInfo : TPaintStruct);
VAR   idx : Integer;
BEGIN
  IF  slip <> nil  THEN
  BEGIN
  row := 1;
  for  idx := 1 to slip^.NumDish  do
    DispDishPrice(Paintdc,idx);
  END;
END;

PROCEDURE  SubWndC.NewTable(nslp : OdrPaperTP);
BEGIN  slip := nslp    END;
```

รูป B9

ลงรายการอาหาร

งานหลักของหน้าต่างเอกสารคือการลงรายการอาหารกับคิดเงิน เมื่อผู้ใช้เลือกรายการอาหารที่เมนู วินโดว์จะส่งสื่อความมาให้แก่กิจกรรมที่เตรียมไว้แล้ว ดังที่เคยกล่าวไว้แล้ว ในรูป A2 (ไมโครฯ ฉบับที่ 82)

กิจกรรมต่าง ๆ เหล่านี้จะเรียกใช้ RegDispDish ในรูป B10 โดยผ่านค่าหมายเลขอาหารของตนมาให้ ซึ่ง RegDispDish จะนำไปเก็บในใบสั่งอาหาร และให้ DispDishPrice นำออกแสดงรายละเอียดดังที่เคยกล่าวแล้ว แต่เนื่องจากมีการแก้โปรแกรมจึงนำมาแสดงครับ

ขอให้สังเกตอีกครับว่า ใน RegDispDish ถ้า slip เป็น nil จะไม่ดำเนินการใด เปรียบเสมือนว่าให้ตัดการติดต่อกับผู้ใช้

คิดยอดรวม

เมื่อลูกค้าชำระเงิน ผู้ใช้จะเลือกรายการ Cash ทำให้วินโดว์ส่งสื่อความ cm_Cash มาทำงานที่กิจกรรม cmCash ในรูป B11

จะตรวจสอบอีกเช่นกันว่า ถ้า slip ไม่เป็น nil และรวมทั้งต้องมีการสั่งอาหารด้วย จึงจะรวมยอดเงินโดยนำหมายเลขอาหาร (คือ dnum) ไปเปิดในตารางราคารวมยอดเป็นยอดรวม ให้ RegDispDish นำออกแสดง

แต่ค่าที่ส่งไปให้เป็นเลขลบ ซึ่งใน RegDispDish จะถือว่าเป็นยอดรวม แสดงรายการเป็น TOTAL และกลับยอดรวมเป็นเลขบวกนำออกแสดง

ให้หน้าต่างคุยกัน

จากที่ออกแบบไว้ว่าเมื่อคิดยอดรวมแล้วให้ยกเลิกใบสั่งอาหาร โดยจะให้คำแนะนำผู้ใช้ว่าให้ปิดหน้าต่างนี้ แต่เราบังคับผู้ใช้ไม่ได้ครับจึงต้องเตรียมการไว้ คือให้ผู้ใช้บันทึกรายการอาหารต่อไปไม่ได้ โดยให้ slip เป็น nil ดังกล่าวมา

แต่การยกเลิกใบสั่งอาหารหน้าต่างเอกสารทำไม่ได้ เพราะเก็บอยู่ในที่เก็บในหน้าต่างงาน ก็ต้องแจ้งให้หน้าต่างงานคือแม่ของตนทราบทางสื่อความ

ในการนี้วินโดว์ได้เตรียมการไว้ให้แล้วคือ ได้จัดฟังก์ชัน SendMessage ไว้ให้ ดังส่วนหัวในรูปที่ 7 โดยที่

- wnd เป็นจ่าหน้าว่าจะให้ส่งสื่อความนี้ให้ใคร ทั้งนี้เพราะสามารถส่งสื่อความไปให้แก่หน้าต่างใดก็ได้ ตามโปรแกรมของผมให้เป็นถึงแม่โดยอ่านค่า HWindow ของหน้าต่างแม่มาให้

- msg คือหมายเลขสื่อความตามที่กำหนดขึ้นในช่วงของ wm (window message) ในโปรแกรมคือตามที่กำหนดไว้เป็น wm_Cashing ในรูป B3

- wparam เป็นข้อความที่จะฝากส่งเป็นเลขจำนวนเต็ม Word ผม

```
(* ---------------- Private SubRoutine ---------------- *)

PROCEDURE  SubWndC.RegDispDish(dish : Word);
VAR   dc : HDC;
BEGIN
  IF  slip <> nil  THEN
  BEGIN
  dc := GetDC(HWindow);
  slip^.AddDish(dish);
  DispDishPrice(dc,slip^.NumDish);
  ReleaseDC(HWindow, dc);
  END;
END;

PROCEDURE  SubWndC.DispDishPrice(adc : HDC; idx : Word);
VAR  dishCa : array[0..40] of Char;
     priceCa : array[0..6] of Char;
     col, dnum : Integer;
BEGIN
  slip^.GetDish(idx,dnum);
  if  dnum < 0  then
    begin  StrCopy(dishCa,'    TOTAL');
           Str(Abs(dnum),priceCa)  end
  else
    begin  StrCopy(dishCa,dish[dnum]);
           Str(price[dnum],priceCa)  end;
  TextOut(adc, 10, row * 20, dishCa, StrLen(dishCa));
  TextOut(adc, 150, row * 20, priceCa, StrLen(priceCa));
  inc(row);
END;
```

รูป B10

```
PROCEDURE  SubWndC.cmCash(var msg : TMessage);
VAR    dix, dnum, total : Integer;
BEGIN
   IF  (slip <> nil) and (slip^.NumDish > 0) THEN
  BEGIN
  total := 0;
  for  dix := 1 to slip^.NumDish  do
    begin  slip^.GetDish(dix, dnum);
           total := total + price[dnum]  end;
  RegDispDish(0 - total);
  if  SendMessage(Parent^.HWindow,wm_Cashing,
           slip^.GetTblNum,0) = 1  then  slip := nil;
  END;
END;
```

รูป B11

```
SendMessage(wnd : HWnd; msg,wparam : Word;
                        lparam : LongInt) : LongInt;

ReplyMessage(reply : LongInt);
```

รูปที่ 7 ส่วนหัวของฟังก์ชัน SentMessage และโพรซีเดอร์ Reply Message

จึงฝากหมายเลขโต๊ะของใบสั่งอาหารที่ต้องการให้ถอนไปให้

- lparam เป็นข้อความที่จะฝากส่ง อีกเช่นกันเป็นเลขจำนวนเต็ม LongInt ผมไม่มีข้อความใดจะส่งอีกจึงให้เป็น 0

- สำหรับค่าของฟังก์ชันเป็น LongInt คือคำตอบจากผู้รับ โดยผมนำมาเป็นเงื่อนไขว่าถ้าเท่ากับ 1 ให้ยกเลิกใบสั่งอาหาร (เพื่อทดสอบการตอบรับสื่อความ)

ตอบรับสื่อความตรง

เมื่อส่งสื่อความด้วย SendMessage แล้ว วินโดว์จะรับไปดำเนินการส่งเช่นเดียวกับสื่อความทั้งหลาย การรับก็เช่นกันคือ ได้กำหนดกิจกรรม wmCashing สนองตอบสื่อความนี้อยู่ในรูป B5

กับได้เขียนโปรแกรมสำหรับกิจกรรมนี้ในรูป B12 ก็เหมือนกับสื่อความโดยทั่ว ๆ ไป ที่เป็นพิเศษคือได้นำค่าใน msg ซึ่งไม่เคยนำมาใช้ มาใช้ในคราวนี้เพราะเก็บข้อความที่ผู้ส่งฝากมาโดยนำหมายเลขโต๊ะไปค้นใบสั่งอาหารในที่เก็บนำมายกเลิก (หากสงสัยให้ดูแบบข้อมูล TMessage ใน Help ครับ)

แต่ที่แตกต่างกันคือเป็นสื่อความตรง ส่งในแบบมีผู้นำส่งและรอรับกลับด้วย โดยมารยาทจึงต้องตอบรับด้วย ReplyMessage และผมฝากค่า 1 ไปให้ ค่านี้จะกลายเป็นค่าของฟังก์ชัน SendMessage ที่ใช้เป็นเงื่อนไขในรูป B11

โปรแกรมที่สมบูรณ์

เรื่องของหน้าต่างงานกับหน้าต่างเอกสารมีเพียงเท่านี้ครับ ส่วนที่เก็บข้อมูลกับการส่งสื่อความยังมีรายละเอียดอีกในการป้อนโปรแกรมเพื่อให้เรื่องเดียวกันอยู่ด้วยกันขอให้ป้อนหรือแก้ของเดิมตามลำดับดังนี้ครับ

B3, B1, B5, B4, B2, B9 ตามด้วยโปรแกรมสนองตอบสื่อความในสกุล SubWndC ของเดิมที่เปลี่ยนชื่อสกุลแล้ว ตามด้วย B11, B10, B6, B7, B12, B8 และ A8 (ไมโครฯ ฉบับที่ 82) คิดว่าคงไม่มีปัญหา หากมีค่อยพึ่งบริการซอฟต์เมล์ครับ

หากไม่ต้องการใช้ที่เก็บข้อมูล

ตามที่กล่าวแต่ต้นว่าโครงงานนี้ไม่ต้องใช้ที่เก็บข้อมูล ผมนำมาใช้เพื่อกล่าวถึงและถ้าไม่ใช้ที่เก็บข้อมูลก็ไม่ต้องส่งสื่อความคือ SendMessage จะดำเนินการได้ดังนี้คือ

เรื่องแรกคือ จะสร้างสกุลให้แก่ใบสั่งอาหารหรือไม่ หากไม่สร้างจะต้องมีกิจกรรมที่จะดำเนินการแทนในสกุล SubWndC ซึ่งก็ไม่มากแต่เพื่อให้ผมอธิบายได้ง่ายจะถือว่าได้สร้างสกุลให้

จะเขียนโปรแกรมสำหรับสกุล SubWndC ในส่วนที่แตกต่างกัน และที่เพิ่มได้ดังรูปที่ 8 คือ

- ใน Init ให้ผ่านค่าหมายเลขหน้าต่างมาแทนใบสั่งอาหาร และการจองหน่วยความจำสำหรับใบสั่งอาหารเป็นหน้าที่ของสกุลนี้

- เพิ่มกิจกรรม Done เพื่อยกเลิกหน่วยความจำที่จองไว้ ดังนั้นเมื่อผู้ใช้ปิดหน้าต่างจะยกเลิกหน่วยความจำนี้ด้วย

- ให้มีแฟล็กเช่นให้ชื่อว่า inuse ให้เป็นจริง เมื่อ Init และเป็นเท็จเมื่อผู้ใช้เลือกรายการ Cash

- เพิ่มกิจกรรม CanClose โดยให้ค่า CanClose เป็น not inuse ทำให้ผู้ใช้ปิดหน้าต่างไม่ได้จนกว่าจะเลือกรายการ Cash แล้ว

- ใน cmCash ไม่มี SendMessage เปลี่ยนเป็นให้ inuse เป็น False แทน

- ในกิจกรรมที่เคยตรวจสอบ slip ให้เปลี่ยนเป็นตรวจสอบ inuse แทน

เมื่อเป็นเช่นนี้สถานะของหน้าต่างกับใบสั่งอาหารจะเป็นไปด้วยกัน คงมีแต่สถานะเก่ากับใหม่เท่านั้น SubWndC จึงมีกิจกรรมและโปรแกรมย่อยน้อยกว่าโดยถือหลักจาก CreateChild ซึ่งจะมีการทำงานเป็น

ถ้าเป็นหน้าต่างเก่าให้ PopSubwnd

ถ้าเป็นหน้าต่างใหม่ให้ CreateNew

```
PROCEDURE  WndProC.cmCashing(var msg : TMessage);
VAR     tnum,idx : Integer;  sptr : OdrPaperTP;
        wptr : SubWndCP;
BEGIN
  tnum := msg.WParam;  OldSlip(tnum,sptr);
  pole^.Delete(sptr);
  ReplyMessage(1);
END;
```

รูป B12

```
(* ------------------  ResTurntC method  ---------------- *)

PROCEDURE  ResTurntC.InitMainWindow;
BEGIN
  MainWindow := New(WndProCP, Init('GOODLUCK RESTAURANT',
                    LoadMenu(HInstance,'RestMenu')));
END;

(* --------------------  MAIN PROGRAM  ----------------- *)

VAR   app :  ResTurntC;

BEGIN
  app.Init('Restaurant');
  app.Run;
  app.Done;
END.
```

รูป B11

```
CONSTRUCTOR SubWndC.Init(apar : PWindowsObject;
                         tnum : Integer);
VAR   atitle : array[0..12] of Char;
      wnme   : array[0..5] of Char;
BEGIN
  tblnum := tnum;  Str(tblnum,wnme);
  StrCat(StrECopy(atitle,'TABLE '),wnme);
  TWindow.Init(apar, atitle);
  New(slip, Init(tblnum);  inuse := True;
END;


PROCEDURE  SubWndC.Done;
BEGIN  Dispose(slip)  END;


FUNCTION  SubWndC.CanClose:
BEGIN   CanClose := not inuse  END;
```

รูปที่ 8 โปรแกรมของบางกิจกรรมเมื่อไม่ใช้ที่เก็บข้อมูล

```
FUNCTION  WndProC.NunChild : Integer;
VAR   cntr : Integer;

  procedure  CountIt(wnd : SubWndCP);
  begin  inc(cntr)   end;

BEGIN
  cntr := 0;  ForEach(@CounIt);
  NumChild := cntr;
END;
```

รูปที่ 9 แสดงการใช้ ForEach

บางกิจกรรมกับลิสต์

มีกิจกรรมที่ดำเนินการกับลิสต์ 2 กิจกรรมที่ผมเห็นว่าน่าจะนำมากล่าวในตอนนี้เพราะตัวอย่างอยู่ใกล้มือ คือ

ForEach

เป็นกิจกรรมที่ให้ดำเนินการกับออปเจ็กต์ทีละออปเจ็กต์ตั้งแต่ต้นลิสต์จนถึงสุดลิสต์ มีทั้งที่ใช้กับที่เก็บข้อมูลอยู่ในสกุล TCollection และกับลิงค์ลิสต์ของแต่ละหน้าต่างแม่ คืออยู่ในสกุล TWindowsObject

มีข้อกำหนดในการใช้เช่นเดียวกับ FirstThat ต่างกันตรง

- ให้โปรแกรมย่อยภายในเป็นโพรซีเดอร์ (แทนที่จะเป็นฟังก์ชัน) เป็นโปรแกรมดำเนินการ
- ForEach จะส่งออปเจ็กต์ที่อยู่ในลิสต์มาให้แก่โปรแกรมดำเนินการทีละออปเจ็กต์ตั้งแต่ต้นลิสต์จนสุดลิสต์ จึงขึ้นอยู่กับเราว่าจะให้ดำเนินการอะไร

เช่นต้องการทับจำนวนหน้าต่างลูกในหน้าต่างแม่จะเขียนโปรแกรมได้ดังรูปที่ 9

LastThat

มีเฉพาะกับที่เก็บข้อมูลคือ ในสกุล TCollection เหมือนกับ FirstThat ทุกประการ แต่เป็นการดำเนินการจากท้ายลิสต์

ที่ผิดในคู่มือ

ขอส่งท้ายด้วยค่าที่ผมป้อนเป็นค่ารายการในรูปที่ 3 ที่เป็นตัวเลข 5 หลักไม่ตรงกันในคู่มือ และใน Help ค่าเหล่านี้ผมลอกมาจากไฟล์ MDIAPP.RES แต่ยังมีค่าอื่นอีกจะดูได้ในไฟล์ C:\TPW-\DOC\WOBJECTS.INT ซึ่งต้องถือว่าเป็นค่าที่ถูกต้องเพราะเป็นค่าที่ให้ไว้จริงครับ